# Learning Outcome

# After completing this module, the student should be able to learn scikit-learn library.

To meet the learning outcome, a student has to complete the following activities

1. Installing sklearn library (2 hrs )
2. Simple linear regression using excel ( 3 hrs )
3. OLS in sklearn (3 hrs)
4. Train-test-split of data in sklearn (3 hrs)
5. Methods of linear regression- fit(), predict(), coeff\_, intercept\_, score() (3 hrs )

# Activity 1

## Aim: Installing sklearn library.

**Learning outcome:** Able to install scikit-learn library.

**Duration:** 2 hours

**List of Hardware/Software requirements:**

1. Laptop/Computer with Windows 10 / Linux OS - Ubuntu 18.04 LTS
2. Jupyter notebook / Google colab
3. Python 3 and above

**Code/Program/Procedure (with comments):**

**Operating System - Windows**

1. Scikit-learn requires Python 3.6+. To check which version of Python you have installed, run the following command:

**python3 --version**

The output should be similar to:

Python 3.8.2

2. If you have a valid Python version you can run the following command to download and install a pre-built binary of scikit-learn:

**pip install scikit-learn**

The following dependencies will be automatically installed along with scikit-learn:

**NumPy 1.13.3+**

**SciPy 0.19.1+**

**Joblib 0.11+**

**threadpoolctl 2.0.0+**

Alternatively, if you already have scikit-learn and/or any of its dependencies are already installed, they can be updated as part of the installation by running the following command:

**pip install -U scikit-learn**

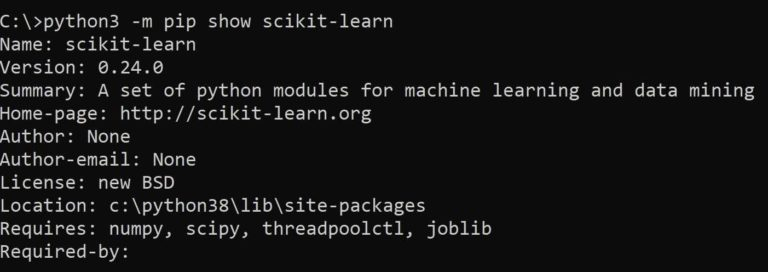
**Operating system – Ubuntu**

**$ sudo apt-get install python3-sklearn python3-sklearn-lib python3-sklearn-doc**

You can verify your Scikit-learn installation with the following command:

**python -m pip show scikit-learn**

**Output/Results snippet:**



**References:**

* <https://www.geeksforgeeks.org/how-to-install-scikit-learn-on-linux/>
* <https://scikit-learn.org/stable/install.html>
* <https://www.activestate.com/resources/quick-reads/how-to-install-scikit-learn/>

# Activity 2

## Aim: Simple linear regression using excel.

**Learning outcome:** Able to learn how linear regression can be done with or without excel specific tool.

**Duration:** 3 hours

**List of Hardware/Software requirements:**

1. Laptop/Computer with Windows 10 / Linux OS - Ubuntu 18.04 LTS
2. Microsoft Excel 2013 and above

**Code/Program/Procedure (with comments):**

Linear regression is a statistical technique/method used to study the relationship between two continuous quantitative variables.

A linear regression line has an equation of the kind: **Y= a + bX;**

Where:

**X** is the explanatory variable,

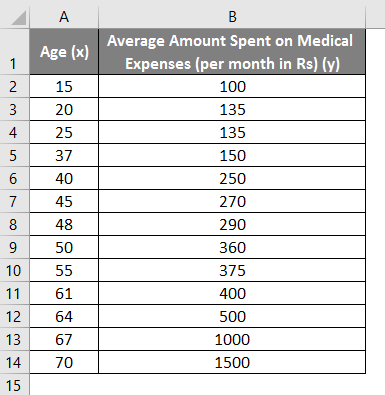
**Y** is the dependent variable,

**b** is the slope of the line,

**a** is the y-intercept (i.e., the value of y when x=0).

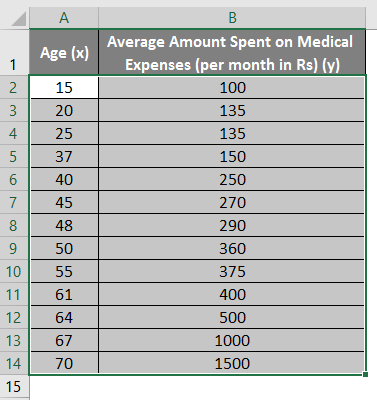
**Method #1 – Scatter Chart with a Trendline**

Let us say we have a dataset of some individuals with their age, bio-mass index (BMI), and the amount spent by them on medical expenses in a month. Now with an insight into the individuals’ characteristics like age and BMI, we wish to find how these variables affect the medical expenses, and hence use these to carry out regression and estimate/predict the average medical expenses for some specific individuals. Let us first see how only age affects medical expenses. Let us see the dataset:

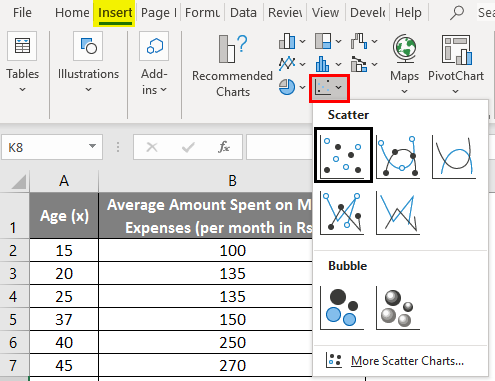


Amount on medical expenses= b\*age + a

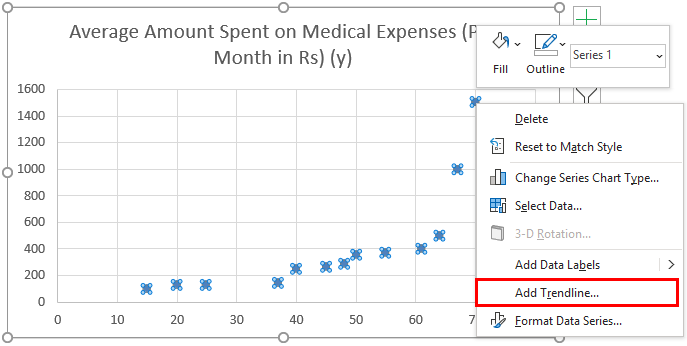
* Select the two columns of the dataset (x and y), including headers.



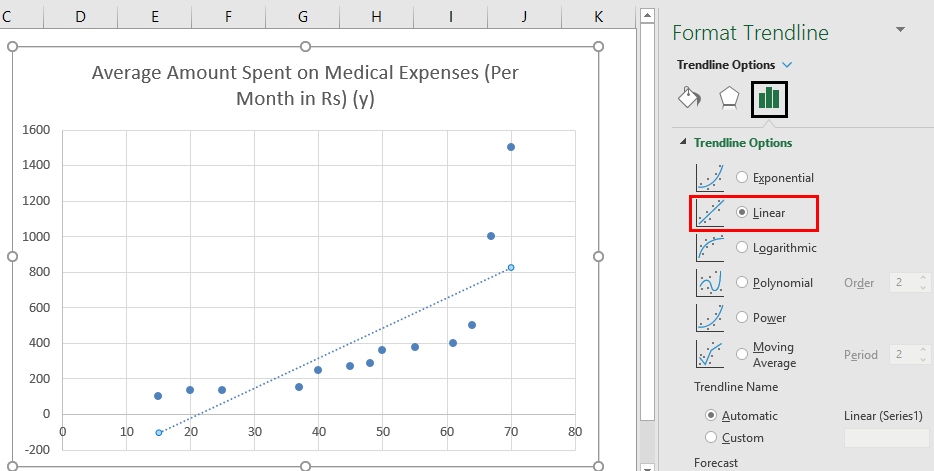
* Click on ‘Insert’ and expand the dropdown for ‘Scatter Chart’ and select ‘Scatter’ thumbnail (first one)



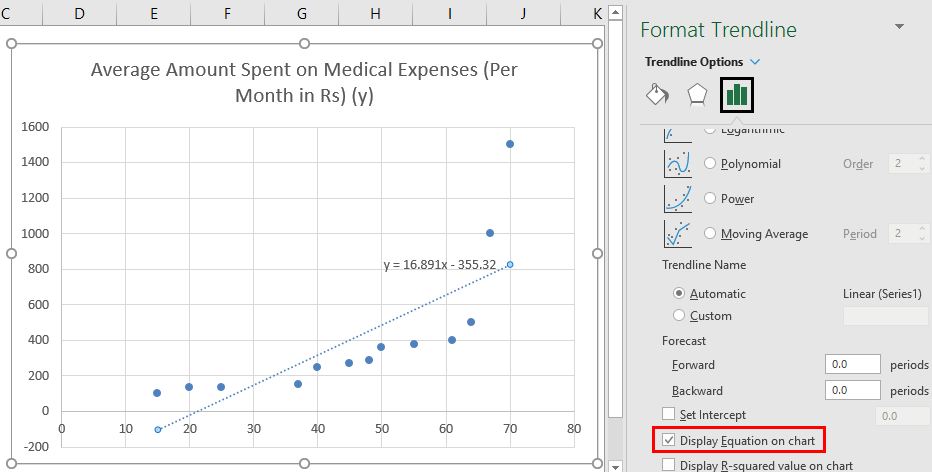
* Now a scatter plot will appear, and we would draw the regression line on this. To do this, right-click on any data point and select ‘Add Trendline.’



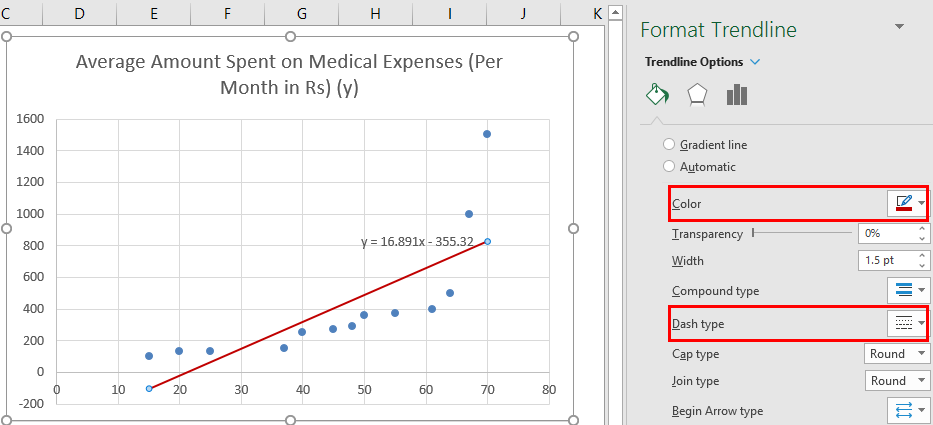
* Now in the ‘Format Trendline’ pane on the right, select ‘Linear Trendline’ and ‘Display Equation on Chart’.



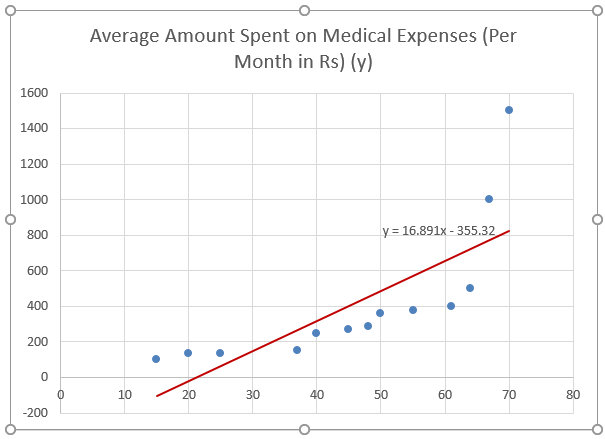
* Select ‘Display Equation on Chart’.



* We can improvise the chart as per our requirements, like adding axes titles, changing the scale, color and line type.

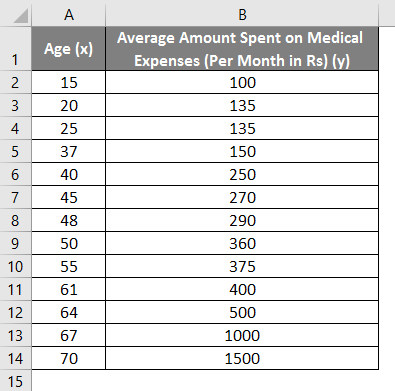


**Output/Results snippet:**

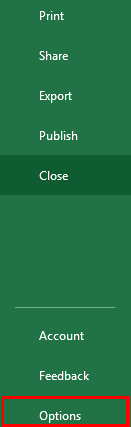


**Method #2 – Analysis ToolPak Add-In Method**

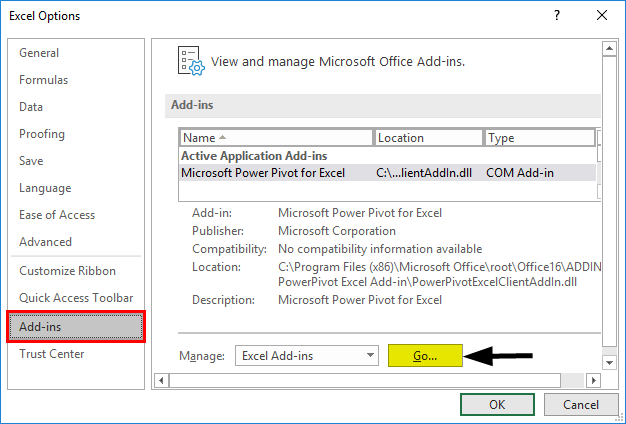
Analysis ToolPak is sometimes not enabled by default, and we need to do it manually. To do so:

****

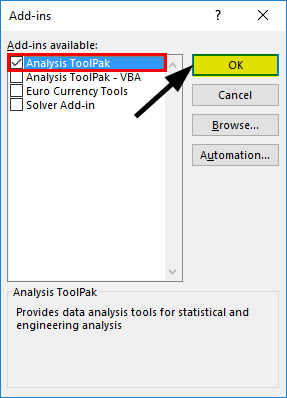
* Click on the ‘File’ menu, after that, click on ‘Options’.



* Select ‘Excel Add-Ins’ in the ‘Manage’ box, and click on ‘Go.’

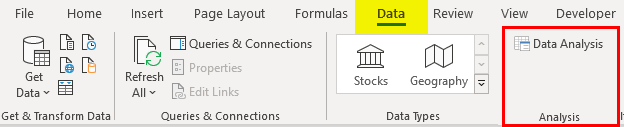


* Select ‘Analysis ToolPak’ -> ‘OK’

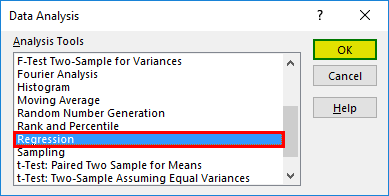


This will add ‘Data Analysis’ tools to the ‘Data’ tab. Now we run the regression analysis:

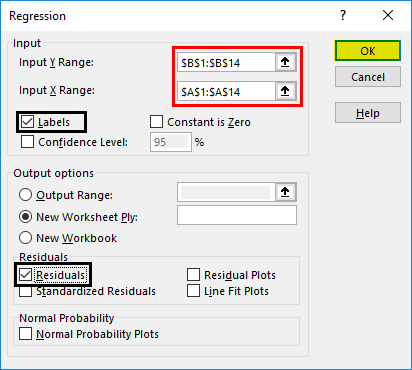
* Click on ‘Data Analysis’ in the ‘Data’ tab



* Select ‘Regression’ -> ‘OK’

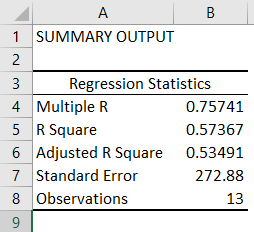


* A regression dialog box will appear. Select the Input Y range and Input X range (medical expenses and age, respectively). In the case of multiple linear regression, we can select more columns of independent variables (like if we wish to see the impact of BMI as well on medical expenses).
* Check the ‘Labels’ box to include headers.
* Choose the desired ‘output’ option.
* Select the ‘residuals’ checkbox and click ‘OK.

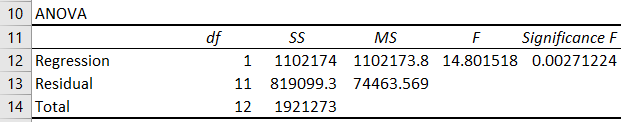


**Output/Results snippet:**

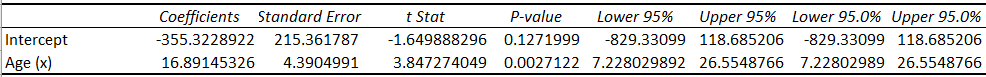
* **Regression Statistics** tells how well the regression equation fits the data:

****

* **Multiple R** is the correlation coefficient that measures the strength of a linear relationship between two variables. It lies between -1 and 1, and its absolute value depicts the relationship strength with a large value indicating a stronger relationship, a low value indicating negative and zero value indicating no relationship.
* **R Square** is the Coefficient of Determination used as an indicator of goodness of fit. It lies between 0 and 1, with a value close to 1 indicating that the model is a good fit. In this case, 0.57=57% of y-values are explained by the x-values.
* **Adjusted R Square** is R Square adjusted for a number of predictors in the case of multiple linear regression.
* **Standard Error** depicts the precision of regression analysis.
* **Observations** depict the number of model observations.
* **Anova** tells the level of variability within the regression model.



* **Coefficients** are the most important part used to build regression equation.

****

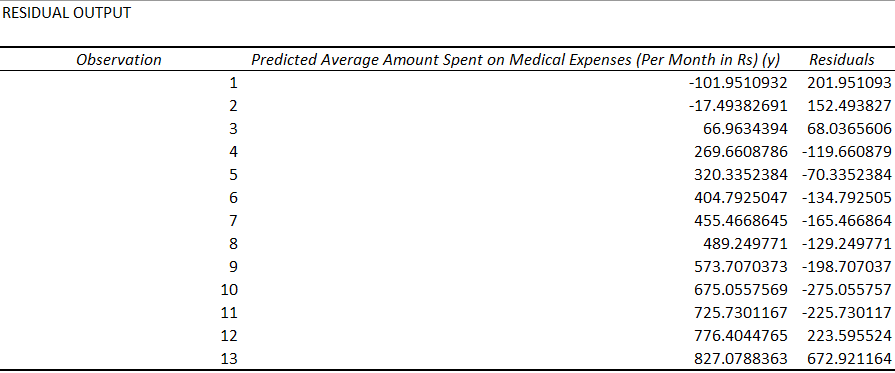
So, our regression equation would be: y= 16.891 x – 355.32. This is the same as that done by method 1 (scatter chart with a trendline).

Now, if we wish to predict average medical expenses when age is 72:

So, y= 16.891 \* 72 -355.32 = 860.832

So, this way, we can predict values of y for any other values of x.

* **Residuals** indicate the difference between actual and predicted values.

****

**References:**

* <https://www.educba.com/linear-regression-in-excel/>
* <https://www.statology.org/simple-linear-regression-excel/>

# Activity 3

## Aim: OLS in sklearn

**Learning outcome:** Able to learn how linear regression can be done with or without excel specific tool.

**Duration:** 3 hours

**List of Hardware/Software requirements:**

1. Laptop/Computer with Windows 10 / Linux OS - Ubuntu 18.04 LTS
2. Jupyter notebook / Google colab
3. Python 3 and above

**Code/Program/Procedure (with comments):**

**Ordinary Least Squares** is a method for finding the linear combination of features that best fits the observed outcome in the following sense.

If the vector of outcomes to be predicted is y, and the explanatory variables form the matrix X, then OLS will find the vector β solving

**minβ|y^ - y|22,**

where **y^ = X β** is the linear prediction.

In sklearn, this is done using **sklearn.linear\_model.LinearRegression**

**Application Context**

OLS should only be applied to regression problems; it is generally unsuitable for classification problems: Contrast

Is an email spam? (Classification)

What is the linear relationship between upvotes depend on the length of answer? (Regression)

**Discovering the Data**

import pandas as pd

dataset\_url = 'https://sealevel-nexus.jpl.nasa.gov/data/ice\_shelf\_dh\_mean\_v1/ice\_shelf\_dh\_mean\_v1\_height.csv'

dataset = pd.read\_csv(dataset\_url)

dataset.head()

#Let’s create x and y vectors.

import numpy as np

# Read CSV into table and get (x, y) pairs.

N = dataset.shape[0] # size of input samples

x = np.array(dataset['Year']).reshape([N, 1])

y = np.array(dataset['All Antarctica']).reshape([N, 1])

points = np.hstack([x, y])

**Creating the Model - Least Squares Estimation**

**Solve the Least Squares Regression by Hand**

# Calculate power series sums.

x0 = np.sum(x\*\*0)

x1 = np.sum(x\*\*1)

x2 = np.sum(x\*\*2)

x3 = np.sum(x\*\*3)

x4 = np.sum(x\*\*4)

yx0 = np.sum(y \* x\*\*0)

yx1 = np.sum(y \* x\*\*1)

yx2 = np.sum(y \* x\*\*2)

# Create 3rd order model matrices.

A = [[x0, x1, x2], [x1, x2, x3], [x2, x3, x4]]

B = [[yx0], [yx1], [yx2]]

**Obtain Model Coefficients**

import numpy.linalg as lin

M = np.matmul(lin.inv(A), B)

The degree-two polynomial coefficients are found as below.

[[-5.48765643e+03],

[ 5.49213398e+00],

[-1.37413749e-03]]

**Simulate the Estimated Curve**

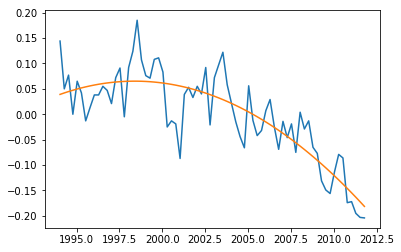
#To visualize the result, we can create y\_estimated

import matplotlib.pyplot as plt

y\_estimated = x\*\*0 \* M[0] + x\*\*1 \* M[1] + x\*\*2 \* M[2]

plt.plot(x, y, x, y\_estimated)

plt.show()

****

**Prediction of Future Values**

y2020 = 2020\*\*0 \* M[0] + 2020\*\*1 \* M[1] + 2020\*\*2 \* M[2]

**RMS Error**

#To see the overall performance of the fit, we can simply take root-mean-square of the error.

rmse = (np.sum((yest - y) \*\*2) / len(y)) \*\* 0.5

**Output/Results snippet:**

The result is 0.047179935281228005.

**References:**

* <http://www.atakansarioglu.com/machine-learning-example-generalized-least-squares-sklearn-scikit-python-hands-on/#solve_the_least_squares_regression_by_hand>
* <https://www.datarobot.com/blog/ordinary-least-squares-in-python/>

# Activity 4

## Aim: Train-test-split of data in sklearn

**Learning outcome:** Able to learn how to Train-test-split of data in sklearn.

**Duration:** 3 hours

**List of Hardware/Software requirements:**

1. Laptop/Computer with Windows 10 / Linux OS - Ubuntu 18.04 LTS
2. Jupyter notebook / Google colab
3. Python 3 and above

**Code/Program/Procedure (with comments):**

**Configuring Test Train Split**

Before splitting the data, you need to know how to configure the train test split percentage.

In most cases, the common split percentages are

**Train: 80%, Test: 20%**

**Train: 67%, Test: 33%**

**Train: 50%, Test: 50%**

**Loading The Dataset**

import numpy as np

from sklearn.datasets import load\_iris

# the iris dataset which has four features Sepal\_length, Sepal\_width, Petal\_length, and Petal\_Width

iris = load\_iris()

x = iris.data

y = iris.target

**Train Test Split Using Sklearn Library**

You can split the dataset into train and test set using the train\_test\_split() method of the sklearn library. It accepts one mandatory parameter.

*–Input Dataset* – It is a sequence of array-like objects of the same size. Allowed inputs are lists, NumPy arrays, scipy-sparse matrices, or pandas data frames.

The Input dataset passed as X and y along with the test\_size = 0.4. It means the data will be split into 60% for training and 40% for testing.

from collections import Counter

from sklearn.model\_selection import train\_test\_split

#Split dataset into train and test

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.4)

print(Counter(y\_train))

print(Counter(y\_test))

**Output/Results snippet:**

Counter({0: 34, 1: 25, 2: 31})

Counter({0: 16, 1: 25, 2: 19})

The train set contains, 34 number of 0 labels, 25 number of 1 labels, and 31 number of 2 labels.

**Train Test Split with Groups**

You can do a train test split with groups using the GroupShuffleSplit() method from the sklearn library.

from sklearn.datasets import load\_iris

from sklearn.model\_selection import GroupShuffleSplit

import pandas as pd

data = load\_iris()

df = pd.DataFrame(data.data, columns=data.feature\_names)

df["target"] = data.target

train\_idx, test\_idx = next(GroupShuffleSplit(test\_size=.20, n\_splits=2, random\_state = 7).split(df, groups=df['target']))

train = df.iloc[train\_idx]

test = df.iloc[test\_idx]

#To display the training set

train.groupby(['target']).count()

**Output/Results snippet:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **sepal length (cm)** | **sepal width (cm)** | **petal length (cm)** | **petal width (cm)** |
| **target** |  |  |  |  |
| 0 | 50 | 50 | 50 | 50 |
| 1 | 50 | 50 | 50 | 50 |

#To print the test dataset count.

test.groupby(['target']).count()

**Output/Results snippet:**

Dataframe will look like

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **sepal length (cm)** | **sepal width (cm)** | **petal length (cm)** | **petal width (cm)** |
| **target** |  |  |  |  |
| 2 | 50 | 50 | 50 | 50 |

**References:**

* <https://www.stackvidhya.com/train-test-split-using-sklearn-in-python/>
* <https://stackabuse.com/scikit-learns-traintestsplit-training-testing-and-validation-sets/>

# Activity 5

## Aim: Methods of linear regression- fit(), predict(), coeff\_, intercept\_, score()

**Learning outcome:** Able to learn different methods of linear regression- fit(), predict(), coeff\_, intercept\_, score()

**Duration:** 3 hours

**List of Hardware/Software requirements:**

1. Laptop/Computer with Windows 10 / Linux OS - Ubuntu 18.04 LTS
2. Jupyter notebook / Google colab
3. Python 3 and above

**Code/Program/Procedure (with comments):**

**Step 1: Import packages and classes**

import numpy as np

from sklearn.linear\_model import LinearRegression

**Step 2: Provide data**

#The inputs (regressors, 𝑥) and output (predictor, 𝑦) should be arrays (the instances of the class numpy.ndarray)

x = np.array([5, 15, 25, 35, 45, 55]).reshape((-1, 1))

y = np.array([5, 20, 14, 32, 22, 38])

>>> print(x)

**Output/Results snippet:**

[[ 5]

[15]

[25]

[35]

[45]

[55]]

>>> print(y)

**Output/Results snippet:**

[ 5 20 14 32 22 38]

**Step 3: Create a model and fit it**

model = LinearRegression()

model.fit(x, y)

model = LinearRegression().fit(x, y)

>>> r\_sq = model.score(x, y)

>>> print('coefficient of determination:', r\_sq)

**Output/Results snippet:**

coefficient of determination: 0.715875613747954

>>> print('intercept:', model.intercept\_)

**Output/Results snippet:**

intercept: 5.633333333333329

>>> print('slope:', model.coef\_)

**Output/Results snippet:**

slope: [0.54]

>>> new\_model = LinearRegression().fit(x, y.reshape((-1, 1)))

>>> print('intercept:', new\_model.intercept\_)

**Output/Results snippet:**

intercept: [5.63333333]

>>> print('slope:', new\_model.coef\_)

**Output/Results snippet:**

slope: [[0.54]]

**Step 5: Predict response**

>>> y\_pred = model.predict(x)

>>> print('predicted response:', y\_pred, sep='\n')

**Output/Results snippet:**

predicted response:

[ 8.33333333 13.73333333 19.13333333 24.53333333 29.93333333 35.33333333]

>>> y\_pred = model.intercept\_ + model.coef\_ \* x

>>> print('predicted response:', y\_pred, sep='\n')

**Output/Results snippet:**

predicted response:

[[ 8.33333333]

[13.73333333]

[19.13333333]

[24.53333333]

[29.93333333]

[35.33333333]]

>>> x\_new = np.arange(5).reshape((-1, 1))

>>> print(x\_new)

**Output/Results snippet:**

[[0]

[1]

[2]

[3]

[4]]

>>> y\_new = model.predict(x\_new)

>>> print(y\_new)

**Output/Results snippet:**

[5.63333333 6.17333333 6.71333333 7.25333333 7.79333333]